A File Structure for The Complex, The Changing and the Indeterminate

Theodor H. Nelson
Vassar College, Poughkeepsie, N.Y. 12601

Reprinted from
Association for Computing Machinery Proceedings of the 20th National Conference
with permission of ACM and the editor-publisher, Lewis Winner.
Minor corrections by the author.

© 1965 By The
Association for Computing Machinery

SUMMARY

THE KINDS OF FILE structures required if we are to use the computer for personal files and as an adjunct to creativity are wholly different in character from those customary in business and scientific data processing. They need to provide the capacity for intricate and idiosyncratic arrangements, total modifiability, undecided alternatives, and thorough internal documentation.

The original idea was to make a file for writers and scientists, much like the personal side of Bush's Memex, that would do the things such people need with the richness they would want. But there are so many possible specific functions that the mind reels. These uses and considerations become so complex that the only answer is a simple and generalized building-block structure, user-oriented and wholly general-purpose.

The resulting file structure is explained and examples of its use are given. It bears generic similarities to list-processing systems but is slower and bigger. It employs zippered lists plus certain facilities for modification and spin-off of variations. This is technically accomplished by index manipulation and text patching, but to the user it acts like a multifarious, polymorphic, many-dimensional, infinite blackboard.

The ramifications of this approach extend well beyond its original concerns, into such places as information retrieval and library science, motion pictures and the programming craft; for it is almost everywhere necessary to deal with deep structural changes in the arrangements of ideas and things.

I want to explain how some ideas developed and what they are. The original problem was to specify a computer system for personal information retrieval and documentation, able to do some rather complicated things in clear and simple ways. The investigation gathered generality, however, and has eventuated in a number of ideas. These are an information structure, a file structure, and a file language, each progressively more complicated. The information structure I call zippered lists; the file structure is the ELF, or Evolutionary List File; and the file language (proposed) is called PRIDE.

In this paper I will explain the original problem. Then I will explain why the problem is not simple, and why the solution (a file structure) must yet be very simple. The file structure suggested here is the Evolutionary List File, to be built of zippered lists. A number of uses will be suggested for such a file, in order to show the breadth of its potential usefulness. Finally, I want to explain the philosophical implications of this approach for information retrieval and data structure in a changing world.

I began this work in 1960, with no help from anybody. Its purpose was to create techniques for handling personal file systems and manuscripts in progress. These two purposes are closely related and not sharply distinct. Many writers and research professionals have files or collections of notes which are tied to manuscripts in progress. Indeed, often personal files shade into manuscripts, and the assembly of textual notes becomes the writing of text without a sharp break.

I knew from my own experiment what can be done for these purposes with card file, notebook, index tabs, edge-punching, file folders, scissors and paste, graphic boards, index-strip frames, Xerox machine and the roll-top desk. My intent was not merely to computerize these tasks but to think out (and eventually program) the dream file: the file system that would have every feature a novelist or absent-minded professor could want, holding everything he wanted in just the complicated way he wanted it held, and handling notes and manuscripts in as subtle and complex ways as he wanted them handled.

Only a few obstacles impede our using computer-based systems for these purposes. These have been high cost, little sense of need, and uncertainty about system design.

The costs are now down considerably. A small computer with mass memory and video-type display now costs $37,000; amortized over time this would cost less than a secretary, and several people could use it around the clock. A larger installation servicing an editorial office or a newspaper morgue, or a dozen scientists or scholars, could cost proportionately less and give more time to each user.

The second obstacle, sense of need, is a matter of fashion. Despite changing economies, it is fashionably believed that computers are possessed only by huge organizations to be used only for vast corporate tasks or intricate scientific calculations. As long as people think that, machines will be brutes and not friends, bureaucrats and not helpmeets. But since (as I will indicate) computers could do the dirty work of personal file and text handling, and do it with richness and subtlety beyond anything we know, there ought to be a sense of need. Unfortunately, there are no ascertainable statistics on the amount of time we waste fussing among papers and mislaying things. Surely half the time spent in writing is spent physically rearranging words and paper and trying to find things already written; if 95% of this time could be saved, it would only take half as long to write something.

The third obstacle, design, is the only substantive one, the one to which this paper speaks.

Let me speak first of the automatic personal filing system. This idea is by no means new. To go back only as far as 1945, Vannevar Bush, in his famous article "As We May Think"[1], described a system of this type. Bush's paper is better remembered for its predictions in the field of information retrieval, as he foresaw the spread and power of automatic document handling and the many new indexing techniques it would necessitate. But note his predictions for personal filing:

> Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and, to coin one at random, "memex" will do. A memex is a device in which an individual stores all his books, records, and communications, and which is

mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.

It consists of a desk, and while it can presumably be operated from a distance, it is primarily the piece of furniture at which he works. On the top are slanting translucent screens, on which material can be projected for convenient reading. There is a keyboard, and sets of buttons and levers. Otherwise it looks like an ordinary desk.

....

A special button transfers him immediately to the first page of the index. Any given book of his library [and presumably other textual material, such as notes] can thus be called up and consulted with far greater facility than if it were taken from a shelf. As he has several projection positions, he can leave one item in position while he calls up another. He can add marginal notes and comments, ..." (1, 106-7)

Understanding that such a machine required new kinds of filing arrangements, Bush stressed his file's ability to store related materials in <u>associative trails</u>, lists or chains of documents joined together.

When the user is building a trail, he names it, inserts the name in his code book, and taps it out on his keyboard. Before him are the two items to be joined, projected onto adjacent viewing positions. At the bottom of each there are a number of blank code spaces, and a pointer is set to indicate one of these on each item. The user taps a single key, and the items are permanently joined.

....

Thereafter, at any time, when one of these items is in view, the other can be instantly recalled merely by tapping a button below the corresponding code space. Moreover, when numerous items have been thus joined together to form a trail, they can be reviewed in turn, rapidly or slowly, by deflecting a lever like that used for turning the pages of a book. It is exactly as though the physical items had been gathered together from widely separated sources and bound together to form a new book. It is more than this, for any item can be joined into numerous trails.

....

Thus he goes, building a trail of many items. Occasionally he inserts a comment of his own, either linking it into the main trail or joining it by a side trail to a particular item. (1, 107)

Two decades later, this machine is still unavailable*.

The hardware is ready. Standard computers can handle huge bodies of written information, storing them on magnetic recording media and displaying their contents on CRT consoles, which far outshine desktop projectors. But no programs, no file software are standing ready to do the intricate filing job (keeping track of associative trails and other structures) that the active scientist or thinker wants and needs. While Wallace[3] reports that the System Development Corporation has found it worthwhile to give its employees certain limited computer facilities for their own filing systems, this is a bare beginning.

Let us consider the other desideratum, manuscript handling. The remarks that follow are intended to apply to all forms of writing, including fiction, philosophy, sermons, news and technical writing.

* The Busn Rapid Selector, which he designed (2), is a powerful microfilm instrument, but it is not suited to idiosyncratic personal uses, nor to evolutionary modification, as described hereunder.

The problems of writing are little understood, even by writers. Systems analysis in this area is scanty; as elsewhere, the best doers may not understand what they do. Although there is considerable anecdote and lore about the different physical manuscript and file techniques of different authors, literary tradition demerits any concern with technical systems as detracting from "creativity." (Conversely, technical people do not always appreciate the difficulty of organizing text, since in technical writing much of the organization and phraseology is given, or appears to be.) But in the computer sciences we are profoundly aware of the importance of systems details, and of the variety of consequences for both quality and quantity of work that result from different systems. Yet to design and evaluate systems for writing, we need to know what the process of writing is.

There are three false or inadequate theories of how writing is properly done. The first is that writing is a matter of inspiration. While inspiration is useful, it is rarely enough in itself. "Writing is 10% inspiration, 90% perspiration," is a common saying. But this leads us to the second false theory, that "writing consists of applying the seat of the pants to the seat of the chair." Insofar as sitting facilitates work, this view seems reasonable, but it also suggests that what is done while sitting is a matter of comparative indifference; probably not.

The third false theory is that all you really need is a good outline, created on prior consideration, and that if the outline is correctly followed the required text will be produced. For most good writers this theory is quite wrong. Rarely does the original outline predict well what headings and sequence will create the effects desired: the balance of emphasis, sequence of interrelating points, texture of insight, rhythm, etc. We may better call the outlining process inductive: certain interrelations appear to the author in the material itself, some at the outset and some as he works. He can only decide which to emphasize, which to use as unifying ideas and principles, and which to slight or delete, by trying. Outlines in general are spurious, made up after the fact by examining the segmentation of a finished work. If a finished work clearly follows an outline, that outline probably has been hammered out of many inspirations, comparisons and tests**.

Between the inspirations, then, and during the sitting, the task of writing is one of rearrangement and reprocessing, and the real outline develops slowly. The original crude or fragmentary texts created at the outset generally undergo many revision processes before they are finished. Intellectually they are pondered, juxtaposed, compared, adapted, transposed, and judged; mechanically they are copied, overwritten with revision markings, rearranged and copied again. This cycle may be repeated many times. The whole grows by trial and error in the processes of arrangement, comparison and retrenchment. By examining and mentally noting many different versions, some whole but most fragmentary, the intertwining and organizing of the final written work gradually takes place***.

Certain things have been done in the area of computer manuscript handling. IBM recently announced its "Administrative Terminal System"[5,6,7,8] which permits the storage of unfinished sections of text in computer memory, permits various modifications by the user, and types up the final draft with page numbers, right justification and headers.

While this is a good thing, its function for manuscripts is cosmetic rather than organizing. Such a system can be used only with textual sections which are already well organized, the visible part of the iceberg. The major and strenuous part of such writing must already have been done.

** I understand that this account is reasonably correct for such writers as Tolstoy, Winston Churchill and Katherine Anne Porter. Those who can stick to a prior outline faithfully, like James Fenimore Cooper, tend to be either hacks or prodigies, and don't need this system.

*** For a poignant, mordant portrayal of the writer's struggle, the reader is directed to Gorey's The Unstrung Harp; or, Mr Earbrass Writes a Novel.

If a writer is really to be helped by an automated system, it ought to do more than retype and transpose: it should stand by him during the early periods of muddled confusion, when his ideas are scraps, fragments, phrases, and contradictory overall designs. And it must help him through to the final draft with every feasible mechanical aid-- making the fragments easy to find, and making easier the tentative sequencing and juxtaposing and comparing.

It was for these two purposes, taken together-- personal filing and manuscript assembly-- that the following specifications were drawn up.

Here were the preliminary specifications of the system: It would provide an up-to-date index of its own contents (supplanting the "code book" suggested by Bush). It would accept large and growing bodies of text and commentary, listed in such complex forms as the user might stipulate. No hierarchical file relations were to be built in; the system would hold any shape imposed on it. It would file texts in any form and arrangement desired-- combining, at will, the functions of the card file, loose-leaf notebook, and so on. It would file under an unlimited number of categories. It would provide for filing in Bush trails. Besides the file entries themselves, it would hold commentaries and explanations connected with them. These annotations would help the writer or scholar keep track of his previous ideas, reactions and plans, often confusingly forgotten.

In addition to these static facilities, the system would have various provisions for change. The user must be able to change both the contents of his file and the way they are arranged. Facilities would be available for the revising and rewording of text. Moreover, changes in the arrangements of the file's component parts should be possible, including changes in sequence, labelling, indexing and comments.

It was also intended that the system would allow index manipulations which we may call dynamic outlining (or dynamic indexing). Dynamic outlining uses the change in one text sequence to guide an automatic change in another text sequence. That is, changing an outline (or an index) changes the sequence of the main text which is linked with it. This would permit a writer to create new drafts with a relatively small amount of effort, not counting rewordings.

However, because it is necessary to examine changes and new arrangements before deciding to use or keep them, the system must not commit the user to a new version until he is ready. Indeed, the system would have to provide spin-off facilities, allowing a draft of a work to be preserved while its successor was created. Consequently the system must be able to hold several-- in fact, many-- different versions of the same sets of materials. Moreover, these alternate versions would remain indexed to one another, so that however he might have changed their sequences, the user could compare their equivalent parts.

Three particular features, then, would be specially adapted to useful change. The system would be able to sustain changes in the bulk and block arrangements of its contents. It would permit dynamic outlining. And it would permit the spin-off of many different drafts, either successors or variants, all to remain within the file for comparison or use as long as needed. These features we may call evolutionary.

The last specification, of course, one that emerged from all the others, was that it should not be complicated.

These were the original desiderata. It was not expected at first that a system for this purpose would have wider scope of application; these jobs seemed to be quite enough. As work continued, however, the structure began to look more simple, powerful and general, and a variety of new possible uses appeared. It became apparent that the system might be suited to many unplanned applications involving multiple categories, text summaries or other parallel documents, complex data structures requiring human attention, and files whose relations would be in continuing change.

Note that in the discussion that follows we will pretend we can simply see into the machine, and not worry for the present about how we can actually see, understand and manipulate these files. These are problems of housekeeping, I/O and display, for which many solutions are possible.

Elements of the ELF

What was required we may call an evolutionary file structure: a file structure that can be shaped into various forms, changed from one arrangement to another in accordance with the user's changing need. It was apparent also that some type of list structure was necessary. Making the file out of lists would allow different categories of personal notes, separate drafts, outlines and master indices all to be handled as lists of some sort; their segments could then be manipulated through automatic handling of index numbers. The resulting file structure I will accordingly call the Evolutionary List File, or ELF, since it is an evolutionary file structure constructed with lists. The system proposed here is not the only ELF possible. It is built upon a specific technique of attaching lists together which has a natural resistance to becoming confused and messy.

As computer-based systems grow in capability and diversity of uses, they tend to become more and more cluttered with niggling complications, hidden passageways, and lurking, detailed interlocks, restrictions, specializations, provisos. These should be forsworn, if possible, in the system under discussion, so that it might be attractive to laymen (including artists and writers) who feel unkindly disposed toward computers. It should readily adapt to their own styles of handling things, imposing few conventions or methods of use. How could this imposition be avoided? And among so many interesting and possible system functions and file relations, how may the users know what connections to make, how may they understand what they are doing, and how may they avoid muddling and losing the things they are working with?

The answer, I think you see, is to choose a very simple structure that can be used and compounded in many different ways. The basic arrangement chosen for these purposes is an information structure I will refer to as zippered lists. (We might call it permutation-invariant one-for-one inter-list entry-linking, but that is not necessary.)

There are only three kinds of things in the zippered-list ELF, with no predetermined relations among them-- no hierarchies, machine-based features or trick exceptions. The system is user-oriented and open-faced, and its clear and simple rules may be adapted to all purposes.

The ELF has three elements: entries, lists and links. An entry is a discrete unit of information designated by the user. It can be a piece of text (long or short), a string of symbols, a picture or a control designation for physical objects or operations.

A list is an ordered set of entries designated by the user. A given entry may be in any number of lists.

A link is a connector, designated by the user, between two particular entries which are in different lists (Figure 1). An entry in one list may be linked to only one entry in another list.

On the left we see two zippered lists. Between the entries of list A and those of B are dashed lines, representing the links between the two lists. On the right is the table of links as it might look to a machine. The machine can read this table from right to left or left to right, finding entries in B that correspond to given entries in A, or vice versa. A change in the sequence of either list, or additions to either list, will not change the links that stand between them. Changes in the link structure will occur only if the user specifically changes the links, or if he destroys entries which are linked to others.

To be technical, then, two lists are zippered if there are any pairwise links between their respective elements, each element is in no more than one link pair, and these links are unaffected by permutation of the lists, remaining affixed to the same pairs of elements. It is not required that the two lists be of the same length, or, even if they are, that all entries have a link to the other list.

The ELF's File Operations

Zippered lists are an information structure; the Evolutionary List File is a file structure. The ELF described in this paper holds its contents exclusively as zippered (or unzippered) lists. But the file structure must also include a set of operations by which it may be modified. These file operations exist for creating, adjusting or removing the entry, list and link, and for manipulating the sequence relation. An ELF is actually any machine which will, on command, carry out the basic operations on entry, list, link and sequence.

Entries. The user may create new entries at any time, putting anything in them that he thinks appropriate. Entries may be combined or divided (unless indivisible, like objects, commands, etc.) Entries may be put in any list, and the same entry may be put in different lists. The user may direct that entries of one list be automatically copied onto another list, without affecting the original list.

Lists. The user may create lists and assign entries to them. He may at will make new copies of lists. He may rearrange the sequence of a list, or copy the list and change the sequence of that copy. Lists may be combined; lists may be cut into sublists.

Links. The user may create links between entries that are in different lists. Any number of legal links may be created, although the upper limit of links between any two lists is determined by the 1-for-1 rule. When an entry or a list is copied into a list, links will remain between parent and daughter entries. Moreover, after a list-copying operation, the daughter list will have the same links to all other lists as does the parent list.

Sequences. The user may put a list in any sequence he wishes. (A copied list will maintain the original sequence until modified.) Sequences may be transferred between lists via the links: if the sequence of A is transferred to B, each entry of A linked to an entry in B takes the sequential position of its linked

entry in B.

No definite meaning is assigned to these entities or operations by the system; the user is free to let them mean anything he likes. A list may be a category, trail, index, dialogue, catalogue or poem, and lists may be assembled into larger structures. The ELF may be thought of as a place; not a machine, but a piece of stationery or office equipment with many little locations which may be rearranged with regard to one another****.

Note that zippered lists generate only one of various possible Evolutionary List Files. Indeed, the description of the file structure given here is in some ways restrictive: the ELF could take a number of other, closely similar forms and still be much the same thing. For example, it would be possible to allow subentries and superentries into the file, to behave and link up like normal entries, even though they contained or were contained in other entries. But the equivalent can be done with the current system. Another possibility would be to allow links other than 1-for-1; these could be modal, the different link-modes having different meanings to the user. Or we might make it an evolutionary network file, allowing any two entries to be connected. Or, besides such general changes in the rules, plausible changes and accessory functions for any purposes could be introduced outside the given file structure, even including modifications and widgets to do some of the same things "more easily."

But to the user such complication might render the system far less handy or perspicuous. The ELF, with its associated techniques as described above, is simple and unified. Many tasks can be handled within the file structure. This means it can be of particular benefit to people who want to learn without complications and use it in ways they understand. For psychological, rather than technical reasons, the system should be lucid and simple. I believe that this ELF best meets these requirements.

Technical Aspects

Since the ELF description above bears some resemblance to the list languages, such as IPL, SLIP, etc., a distinction should be drawn. These list languages[9] are particularly suited to processing data, fast and iteratively, whose elements are manipulable in Newell-Shaw-Simon lists. Essentially they may be thought of as organizations of memory which facilitate sequential operations on unpredictably branching or hierarchical data. These data may change far too quickly for human intervention. Evolutionary file structures, and the ELF in particular, are designed to be changed piecemeal by a human individual. While it might be convenient to program an ELF in one of these languages, the low speed at which user file commands need to be executed makes such high-powered implementation unnecessary; the main problem is to keep track of the file's arrangements, not to perform computation on its contents. Although work has been done to accommodate the list-language approach to larger chunks of material than usual[10], the things people will want to put into an ELF will typically be too big for core memory.

The ELF does in fact share some of the problems of the list languages: not available-storage accounting or garbage collection (concerns associated with organization of fast memory for processing, which may be avoided at slower speeds), but the problems of checkout for disposal (what other lists is an entry on?) and list naming. The former problem is rather straightforwardly solved[11, p. 164]; the latter is complicated in ways we cannot go into here.

**** An ELF might even be constructed out of cards, blocks, sticks and strings, using techniques of puppetry, but this would not be a convenient object.

The ELF appears to be closest, topologically and in other organizing features, to the Multilist system described by Prywes and Gray[12]. Like that system, it permits putting entries in many different lists at once. However, in current intent[13] that system is firmly hierarchical, and thus somewhat removed from the ELF's scope of application. Another closely related system is the Integrated Data Store of Bachman[14,15,16,17,18]; this is intended as a hardware-software system for disc I/O and storage arrangement, but in its details it seems the ELF's close relative. Each of these systems has a connection logic that might be feasible as a basis for an ELF different from this one. Or, either might prove a convenient programming base for the implementation of this file structure.

Another obvious technical question must be considered. How can the ELF allow "unlimited" copies of entries and lists? By patching techniques, of course. Variant entries and lists can take virtually no space, being modification data plus pointers to the original. When a modified version of a list or entry is created, the machine patches the original with the changes necessary to make the modified version. (Figure 2)

USES

In the discussion that follows, we will examine various possible applications of zippered lists and the ELF, and postpone discussing the file language they require. Finally we will return to this problem, and describe the file language PRIDE whose additional features are needed to adapt the ELF for the uses originally discussed.

By assigning entries to lists, the ELF may be used as a glorified card file, with separate lists used for categories, trails, etc. This permits extensive cross-indexing by the assignment of one entry to different lists. It permits subsets and sub-sequences for any use to be held apart and examined without disturbing the lists from which they have been drawn, by copying them onto other, new lists. The ELF permits the filing of historical trails or associative (Bush) trails through documents, business correspondence, belles-lettres, case law, treaties, scholarly fields and history, and the mixture of trail with categorical filing.

These are the simple uses; the compound uses are much more interesting. But since we cannot intuitively fit every possible conceptual relationship into zippered lists, imaginative use is necessary. Remember that there is no correct way to use the system. Given its structure, the user may figure out any method useful to him. A number of different arrangements can be constructed in the ELF, using only the basic elements of entry, list and link. Zippered lists may be assembled into rectangular arrays, lattices and more intricate configurations. These assemblies of lists may be assigned meaning in combination by the user, and the system will permit them to be stored, displayed, taken apart for examination, and corrected, updated, or modified.

By using such combining arrangements on lists composed of text, the file can be self-documenting, with all labelling and documentation kept integrally within the file structure. It is thus possible to incorporate, in a body of information filed in the ELF, various levels of index, summary, explanation and commentary. Many useful ways of listing and linking such documentation are possible. In Figure 3 we see some of the ways that documentary lists may be linked together. The lists shown are outline, suboutline, draft, subdraft, summary, commentary and source list. These are not all the possible types of documentary lists; for example, "footnotes" are omitted. The ELF will permit any number of these documentary lists;

observe that they can be built on one another, and indefinitely compounded. The system will have no trouble accepting a commentary on a commentary on a subdraft of an outline for a variant list of source materials.

Figure 3 shows also how two lists may contain some of the same entries. The dashed line represents linkage between entries, the solid line shows that both lists contain the same entry. This may be useful for creating alternate versions, or, as in this example, the lists containing the same entry may have different purposes. Here, for instance, an entry in the summary is also to be found in the main draft.

This self-documentation feature permits any string of text in the ELF, long or short, to be annotated or footnoted for scholarly or other purposes. Such marginalia can be temporary or permanent, for the private memoranda of an individual or for communication among different persons using the file.

In a like manner, the ELF is capable of storing many texts in parallel, if they are equivalent or linked in some way. For example, instruction manuals for different models ot the same machine may be kept in the file as linked lists, and referred to when machines are to be compared, used or fixed. This is of special use to repairmen, project managers and technical writers.

Moreover, the ELF's cross-sequencing feature -- the fact that links ignore permutations-- permits the collation of very different cognate textual materials for comparison and understanding. In law, this would help in comparing statutes (or whole legal systems); in literature, variorum editions and parodies. Thus such bodies as the Interpreter's Bible and a Total Shakespeare (incorporating Folios, bowdlerizations, satires and all critical commentary) could be assembled for study.

Let me try to illustrate the possible comprehensiveness and versatility of this file structure as applied to texts. Figure 4 shows the different arrangements that might be used by one man-- in this case an historian writing a book-- to assemble and integrate his intellectual and professional concerns. Although it is impossible to show the links between all the separate entries of these lists-- the entries are not themselves discernible in this drawing-- it is possible to note the kinds of links between lists. A thin line between lists shows that some links exist; a solid line indicates that some entries of both lists are the same.

Perhaps this looks complicated. In fact, each of the connectors shows an indexing of one body of information to another; this user may query his file in any direction along these links, and look up the parts of one list which are related to parts of another. Therefore the lines mean knowledge and order. Note that in such uses it is the man's job to draw the connections, not the machine's. The machine is a repository and not a judge.

The ELF may be an aid to the mind in creative tasks, allowing the user to compare arrangements and alternatives with some prior ideal. This is helpful in planning nonlinear assemblages (museum exhibits, casting for a play,) or linear constructions of any kind. Such linear constructions include not only written texts; they can be any complicated sequences of things, such as motion pictures (in the editing stage) and computer programs.

Indeed, computer programming with an on-line display and the ELF would have a number of advantages. Instructions might be interleaved indefinitely without

resorting to tiny writing. Moreover, the programmer could keep up work on several variant approaches and versions at the same time, and easily document their overall features, their relations to one another and their corresponding parts. Adding a load-and-go compiler would create a self-documenting programming scratchpad.

The natural shape of information, too, may call for the ELF. For instance, sections of information often arrange themselves naturally in a lattice structure, whose strands need to be separately examined, pondered or tested. Such lattices include PERT networks, programmed instruction sequences, history books and genealogical records. (The ELF can handle genealogical source documentation and its original text as well.) Indeed, any informational networks that require storage, handling and consideration will fit the ELF; a feature that could have applications in plant layout, social psychology, contingency planning, circuit design and itineraries.

The ELF may, through its mutability, its expansibility, and its self-documentation features, aid in the integration, understanding and channeling of ideas and problems that will not yield to ordinary analysis or customary reductions; for instance, the contingencies of planning, which are only partially Boolean. Often the reason for a so-called Grand Strategy in a setting is that we cannot keep track of the interrelations of particular contingencies. The ELF could help us understand the interrelations of possibilities, consequences, and strategic options. In a logically similar case, evaluating espionage, it might help trace consistencies and contradictions among reports from different spies.

The use of an ELF as the basis for a management information system is not inconceivable. Its evolutionary capability would provide a smooth transition from the prior systems, phasing out old paperwork forms and information channels piecemeal. Beginning with conventional accounting arrays and information flow, and moving through discrete evolutionary steps, the ELF might help restructure an entire corporate system. Numerical subroutining could permit the system to encompass all bookkeeping. The addresses of all transaction papers, zippered to lists of their dates and contents, would aid in controlling shipments, inventory and cash. The ELF's cross-sequencing feature could be put to concrete uses, helping to rearrange warehouses (and the company library) by directing the printout of new labels to guide physical rearrangement. Inventories, property numbers and patents could be so catalogued and recatalogued in the ELF. Legal documents, correspondence, company facts and history could be indexed or filed in historical and category trails. And upper management could add private annotations to the public statements, reports and research of both the organization and its competitors, with amendments, qualifications, and inside dope.

PRIDE

While the ELF as described is expected to be general and useful, the original purposes described at the beginning of this paper call for certain further provisions. Now I would like to describe a desirable file and information handling language that will meet these needs, called the PRIDE (Personalized Retrieval, Indexing, and Documentation Evolutionary) System. Its purpose is to facilitate the use of an ELF. The system described is not yet implemented, nor even fully specified, but let us speak as though it is.

PRIDE includes the ELF operations. However, for safety and convenience nearly every operation has an inverse. The user must be permitted, given a list of what he has done recently, to undo it. It follows that "destroy" instructions must

fail safe; if given accidentally, they are to be revocable. For safety's sake, it should take several steps to throw a thing away completely. An important option would permit the user to retrace chronologically everything he does on the system.

Most of PRIDE's applications will involve text handling, either as a primary purpose or in the documentation of some other task. Hence a number of features exist for convenient text usage. Text handling commands (for modifying entries) include the equivalents of standard proofreader's marks for insertion, deletion and switching of sections.

Also for text usage and user comfort, there are certain system non-restrictions. There is no practical restriction on the length of an input entry, and it need follow only the most trivial format conventions. In addition, the machine will interrupt any other PRIDE function to receive input text (inspiration mode). It is necessary that entries of unspecified length be acceptable to the system without fuss or warning. PRIDE does not stipulate fixed record lengths, either for input or storage; any such restrictions would have a psychologically cramping effect. There is no reason the system cannot appear to the user to have no fixed or standard unit lengths; the machine's operating units and sections should not concern him.

Ideally, neither the length of entries, the number of lists, or any other parameter of a file is restricted by anything but the absolute size of all memory. This is a difficult requirement for the programmer. Routinely, however, the system should be able to accept entries thousands of characters long, accept hundreds of entries to a list, and accept hundreds of lists in the file. Otherwise, extraneous consideration by the user of whether there's room to add material or try out an offshoot begins to interfere with the system's use.

Although I have avoided discussing the means by which the user sees his file, PRIDE must, of course, have functions and commands for this purpose. For a CRT these include quick lookup schemes[19], preferably with moving menus and means of readily changing the hierarchy of lookup structure; as well as visual cuing and mnemonic formats, including cursor maneuvers, overlays and animated wipes and other transitions. But such glamorous features do not reduce the challenge or worth of working through a line printer, or seeking to make the system useful under a batch-processing monitor.

Many instructions aside from those already mentioned will be needed by the user; particular applications will require such operations as text lookup and integer arithmetic. And surely all the uses of the system have not been anticipated. Hence a subroutining facility is to be available, reaching to assembly language or opening into the machine's other languages. This could be used for processing the file's contents (e.g., numbers or character strings), or for creating more convenient combined operations out of the different operations dealing with file structure, input-output and text.

PRIDE is one possible way to make an ELF, or any evolutionary file structure, useful. PRIDE would be a foreground, free-standing language with the primary mission of handling files and manuscripts, as discussed at the beginning, and secondary applications in ordering and documenting other kinds of complex information. Its major use would presumably be in connection with time-shared display and information systems. But such a language is only one suggestion. Actually, there is not much reason that the ELF could not be made a standard file structure for all purposes; unused capabilities would not intrude, but would still be there if unex-

pectedly wanted. ELF systems could be built into the file capabilities of general utility software. The actual computation involved is relatively trivial, and the ELF could easily be incorporated into I/O routines or data channel languages. Even small-scale hardware implementations are not unthinkable; a control box between a typewriter and a tape recorder, for instance.

All these applications depend, of course, on the system's being actually useful, which is an empirical question. A number of possible applications have been mentioned. But, except as a crutch to man's fallible mind, is there any reason to suppose that the system has any general applicability in principle?

Philosophy

As "philosophy" I want to speak of two major things. First, complex file structures (like the ELF) make possible the creation of complex and significant new media, the hypertext and hyperfilm. Second, evolutionary file structures (like the ELF) make it possible to keep track of things that have been changing, without our awareness, all along. These include the major categories of human thought, which will go on changing.

Systems of paper have grave limitations for either organizing or presenting ideas. A book is never perfectly suited to the reader; one reader is bored, another confused by the same pages. No system of paper-- book or programmed text-- can adapt very far to the interests or needs of a particular reader or student.

However, with the computer-driven display and mass memory, it has become possible to create a new, readable medium, for education and enjoyment, that will let the reader find his level, suit his taste, and find the parts that take on special meaning for him, as instruction or entertainment.

Let me introduce the word "hypertext"***** to mean a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper. It may contain summaries, or maps of its contents and their interrelations; it may contain annotations, additions and footnotes from scholars who have examined it. Let me suggest that such an object and system, properly designed and administered, could have great potential for education, increasing the student's range of choices, his sense of freedom, his motivation, and his intellectual grasp******. Such a system could grow indefinitely, gradually including more and more of the world's written knowledge. However, its internal file structure would have to be built to accept growth, change and complex informational arrangements. The ELF is such a file structure.

Films, sound recordings, and video recordings are also linear strings, basically for mechanical reasons. But these, too, can now be arranged as non-linear systems-- for instance, lattices-- for editing purposes, or for display with different emphasis. (This would naturally require computer control, using the ELF or a related system, and various cartridge or re-recording devices.) The hyperfilm-- a browsable or vari-sequenced movie-- is only one of the possible hypermedia that require our attention.

So much for what we can create afresh with this structure. What about the things that have already been around awhile?

The physical universe is not all that decays. So do abstractions and categories. Human ideas, science, scholarship and language are constantly collapsing

***** The sense of "hyper-" used here connotes extension and generality; cf. "hyperspace." The criterion for this prefix is the inability of these objects to be comprised sensibly into linear media, like the text string, or even media of somewhat higher complexity. The ELF is a hyperfile.

****** I will discuss this idea at length elsewhere.

and unfolding. Any field, and the corpus of all fields, is a bundle of relationships subject to all kinds of twists, inversions, involutions and rearrangement: these changes are frequent but unpredictable. Recall that computers, once a branch of mathematics, are now their own field (but the development of fluid logic indicates a possible merger with the art of wind instruments). Social relations, psycholinguistics and psychonomics are new fields, even though they rest on no special discoveries; political economy, natural history and social ethics are gone. Within a given area, too, the subheadings of importance are in constant flux. In the social sciences, for instance, the topic headings of the nineteen-thirties now sound quaint.

While the disappearance and up-ending of categories and subjects may be erratic, it never stops; and the meaning of this for information retrieval should be clear. Last week's categories, perhaps last night's field, may be gone today. To the extent that information retrieval is concerned with seeking true or ideal or permanent codes and categories-- and even the most sophisticated "role indicator" syntaxes are a form of this endeavor-- to this extent, information retrieval seems to me to be fundamentally mistaken. The categories are chimerical (or temporal) and our categorization systems must evolve as they do. Information systems must have built in the capacity to accept the new categorization systems as they evolve from, or outside, the framework of the old. Not just the new material, but the capacity for new arrangements and indefinite rearrangements of the old, must be possible. In this light, the ELF, indefinitely revisible and unperturbed by changes in overall structural relations, offers some promise.

There is, then, a general rationale. I believe that such a system as the ELF actually ties in better than anything previously used with the actual processes by which thought is progressively organized, whether into stories or hypertext or library categories. Thus it may help integrate, for human understanding, bodies of material so diversely connected that they could not be untangled by the unaided mind. For both logistic and psychological reasons it should be an important adjunct to imaginative, integrating and creative enterprises. It is useful where relationships are unclear; where contingencies and tasks are undefined and unpredictable; where the structures or final outcome it must represent are not yet fully known; where we do not know the file's ultimate arrangement; where we do not know what parts of the file are most important; or where things are in permanent and unpredictable flux. Perhaps this includes more places than we think. And perhaps here, as in biology, the only ultimate structure is change itself.

CONCLUSION

This paper has proposed a different kind of structure for handling information.

Essentially it is a file with certain storage provisions which, combined, permit the file's contents to be arranged any-which-way, and in any number of ways at once. A set of manipulation functions permits making changes or keeping track of developments. The file is capable of maintaining many different arrangements at the same time, many of which may be dormant. This makes ordinary measures of efficiency inappropriate; as with high fidelity music systems, enrichment is derived from the lavish use of surplus capacity.

The key ideas of the system are the inter-linking of different lists, regardless of sequence or additions; the re-configurable character of a list complex into any humanly conceivable forms; and the ability to make copies of a whole list, or

list complex-- in proliferation, at will-- to record its sequence, contents or arrangement at a given moment. The Evolutionary List File is a member of the class of evolutionary file structures; and its particular advantages are thought to be psychological, not technical. Despite this file's adaptability to complex purposes, it has the advantage of being conceptually very simple. Its structure is complete, closed, and unified as a concept. This is its psychological virtue. Its use can be easily taught to people who do not understand computers. We can use it to try out combinations that interest us, to make alternatives clear in their details and relationships, to keep track of developments as they occur, to sketch things we know, like or currently require; and it will stand by for modifications. It can be extended for all sorts of purposes, and implemented or incorporated in any programming language.

There are probably various possible file structures that will be useful in aiding creative thought. This one operates, as it were, on lists that hook together sideways, and their copies. There may be many more.

REFERENCES

1. Bush, Vannevar, "As We May Think." The Atlantic Monthly, 176:1 (July, 1945), 101-108.
2. Hirsch, Phil, "The Bush Rapid Selector." Datamation, June 1965, 56-7.
3. Wallace, Everett N., "Experience with EDP Support of Individuals' File Maintenance." Parameters of Information Science: Proceedings of the American Documentation Institute, v.I (American Documentation Institute, 1964), 259-261.
4. Gorey, Edward, The Unstrung Harp; or, Mr Earbrass Writes a Novel. Duell, Sloan and Pearce, N.Y., 1953.
5. International Business Machines Data Processing Division, "The IBM Administrative Terminal System." Company brochure 520-1146.
6. International Business Machines Technical Publications Department, "1440-/1460 Administrative Terminal System Application Description." IBM, White Plains, New York.
7. Timberlake, W.D., "Administrative Terminal System" (abstract.) STWP Proceedings, May 1965, no page number. New York: Society of Technical Writers and Publishers, 1965.
8. Farrell, Austin C., "Evolution of Automated Writing." STWP Proceedings, May 1965, no page numbers. New York: Society of Technical Writers and Publishers, 1965.
9. Bobrow, Daniel G. and Bertram Raphael, "A Comparison of List-Processing Languages." Comm. ACM 7:4 (April, 1964), 231-240.
10. Comfort, W.T., "Multiword List Items." Comm. ACM 7:6 (June, 1964), 357-362.
11. Weizenbaum, J., "Knotted List Structures." Comm. ACM 5:3 (March, 1962), 161-165.
12. Prywes, N.S. and H.J. Gray, "The Multilist System for Real Time Storage and Retrieval." Proceedings of IFIP Conference, 1962, 112-116.
13. Prywes, N.S., "Interim Technical Report: The Organization of Files for Command and Control." Moore School of Engineering, March, 1964.
14. Bachman, C.W. and S.B. Williams, "The Integrated Data Store-- A General Purpose Programming System for Random Access Memories." AFIPS Conference Proceedings, v. 26, 411-422. Spartan Books, 1964.
15. General Electric Computer Department, "I-D-S." Company brochure CPB-425.
16. General Electric Computer Department, "Introduction to Integrated Data Store." Company brochure CPB-1040, 1965.
17. Bachman, Charles W., "Software for Random Access Processing." Datamation, April 1965.
18. General Electric Computer Department, "Integrated Data Store: New General Electric Technique for Organizing Business Data." Company publication, January 1965.
19. Corbin, Harold S., and George J. Stock, "On-Line Querying via a Display Console." Fourth National Symposium on Information Display: Technical Session Proceedings, 127-154. Society for Information Display, Washington, 1964.

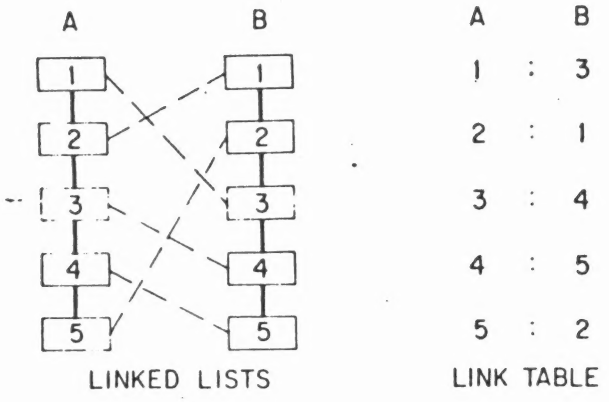

***FIGURE 1*—Zippered lists: 1-for-1 links between entries are invariant under list permutation.**

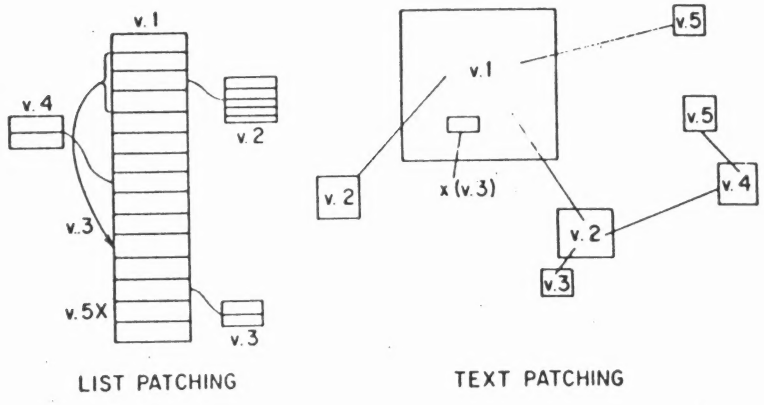

***FIGURE 2*—Spinoff of variants: extra versions need little space.**

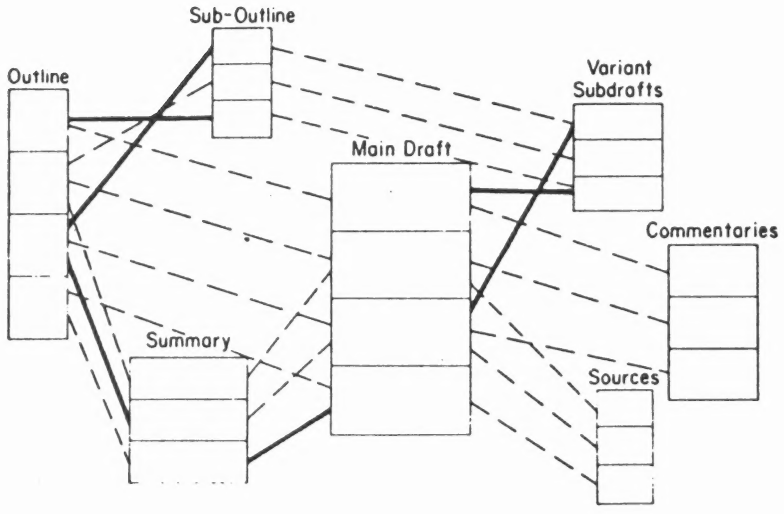

***FIGURE 3*— All levels of documentation may be contained in the ELF. (Heavy lines indicate that linked entries are identical.)**

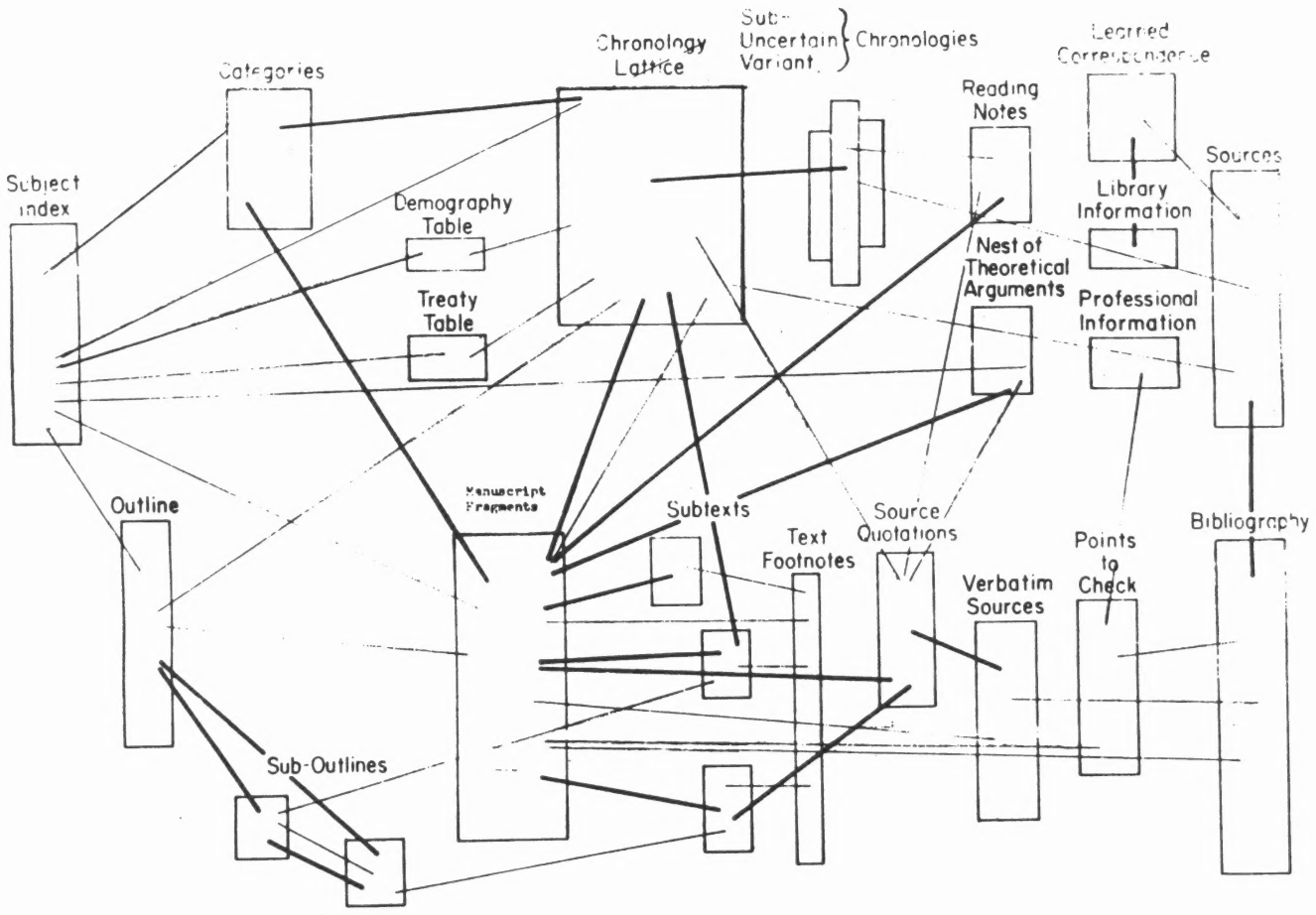

FIGURE 4— ELF's capacity for total filing: hypothetical use by historian. (A thin line indicates the presence of links; a heavy line indicates that some linked entries are identical.)